САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Прикладная математика и информатика
Исследование операций и принятие решений в задачах оптимизации,
управления и экономики

Анохин Иван Александрович

Векторные булевы функции высокой нелинейности

Бакалаврская работа

Научный руководитель:
к. ф.-м. н., доцент Агафонова И. В.

Рецензент:
к. ф.-м. н., доцент Григорьева Н. С.

Санкт-Петербург
2016

SAINT PETERSBURG STATE UNIVERSITY

Applied Mathematics and Computer Science

The Department of Operation Research and Decision Making in Optimisation,
Control and Economics Problems

Anokhin Ivan

Vector Boolean functions of high nonlinearity

Bachelor's Thesis

Scientific supervisor:
Docent Agafonova I. V.

Reviewer:
Docent Grigorieva N. S.

Saint Petersburg
2016

Содержание

Введение

Векторные булевы функции являются одним из ключевых компонент в криптосистемах. Для того чтобы обеспечить устойчивость против криптографических атак, векторная булева функция должна обладать определенными свойствами. Из набора требуемых свойств нас будут интересовать высокая нелинейность и сбалансированность. Данные свойства мы будем требовать от функций как в отдельности, так и совместно.

В данной работе мы опишем и оценим работу нескольких эвристических методов поиска хороших векторных булевых функций: генетический алгоритм, алгоритм имитации отжига, алгоритм восхождения на вершину. Большинство этих методов уже применялось для поиска функций с высокой нелинейностью и сбалансированностью, однако, насколько нам известно, никто не проводил сравнительные анализ их работы.

Кроме того, найдем верхние и нижние границы максимальной нелинейности булевых функций, чтобы оценивать работу алгоритмов не только относительно друг друга, но и относительно максимально возможного значения нелинейности.

Поставим следующие цели и задачи:

1. Построить и уточнить границы максимальной нелинейности;
2. Найти векторные булевы функции с нелинейностью, приближающейся к границам максимальной нелинейности;
3. Найти сбалансированные векторные булевы функции с нелинейностью, приближающейся к границам максимальной нелинейности;
4. Запрограммировать и сопоставить методы, способствующие достижению этой цели.

1 Постановка задачи

В данной главе сформулируем две задачи — это поиск векторных булевых функций, обладающих высокой нелинейностью, и поиск векторных булевых функций, обладающих высокой нелинейностью и сбалансированностью. Эти задачи можно поставить как экстремальные задачи с/без ограничения на сбалансированность соответственно. Для этого определим несколько понятий:

Определение 1. *Булева функция от n переменных – отображение* $V_n \to \mathbb{F}_2$,
где $\mathbb{F}_2$ *— поле из двух элементов, нуля и единицы,* V_n *— векторное пространство над полем* $\mathbb{F}_2$, *состоящее из* 0-1 *векторов длины n с операциями покомпонентного сложения по модулю два и умножения на скаляр.*

Теперь мы можем дать определение векторной булевой функции, которую также будем называть (n, m)-функцией, где n соответствует количеству входных параметров, а m — выходных.

Определение 2. *Векторной булевой функцией или булевой* (n, m)*-функцией от n переменных называется упорядоченный набор функций* $F(x) := (f_1(x), f_2(x), ..., f_m(x))$, *где* $x = (x_1, x_2, ..., x_n)$ *и* $f_i(x)$ *— булевы функции,* $i = 1, ..., m$

Векторные булевы функции в дальнейшем будем стараться обозначать заглавными латинскими буквами, одномерные булевы функции строчными.

Любое число $u \in 0 : 2^n - 1$ посредством последовательного деления на два можно единственным образом представить в виде

$$u = u_{n-1}2^{n-1} + u_{n-2}2^{n-2} + ... + u_1 2 + u_0,$$

где все коэффициенты u_i равны нулю или единице.

Поэтому часто вместо вектора из нулей и единиц длины n будем писать его числовое представление. Например, 5-битным представлением числа 5 является $(0, 0, 1, 0, 1)$. Количество битов, с помощью которого число представляется в двоичной записи, обычно будет ясно из контекста.

Аргументы булевой функции от n переменных принимают конечное число значений, что позволяет представить одномерную булеву функцию в виде вектора значений длины 2^n из нулей и единиц $(f(0), f(1), ..., f(2^n - 2), f(2^n - 1))$ и векторную (n, m)-функцию в виде матрицы $\begin{pmatrix} f_1(0), f_1(1), ..., f_1(2^n - 1) \\ f_2(0), f_2(1), ..., f_2(2^n - 1) \\ ... \\ f_m(0), f_m(1), ..., f_m(2^n - 1) \end{pmatrix}$ размерности $m \times 2^n$ соответственно. Поэтому пространство булевых функций будем обозначать V_{2^n}, пространство векторных булевых (n, m)-функций — $V_{2^n}^m$.

В поле $\mathbb{F}_2$ операцию сложения по модулю два будем обозначать $\oplus$, умножение a и b — ab. Покомпонентное сложение по модулю два двух векторов из V_n также будем обозначать $\oplus$. Всякий раз из контекста будет ясно, к элементам какого пространства применяется операция.

Для определения понятия нелинейности нам понадобится расстояние Хемминга:

Определение 3. *Расстоянием Хемминга $d(f, g)$ между булевыми функциями f и g от n переменных называется число позиций в векторах значений, в которых функции различны.*

Для одномерной булевой функции нелинейность определяется следующим образом:

Определение 4. *Нелинейность булевой функции $f(x_1, ..., x_n)$ от n переменных это* $N_f := \min\limits_{(a_0,...,a_n)\in V_{n+1}} d(f(x_1, ..., x_n), a_0 \oplus a_1x_1 \oplus ... \oplus a_nx_n)$, *где $d(\cdot, \cdot)$ — расстояние Хемминга.*

Расширим понятие нелинейности на класс векторных булевых функций. Для этого на основе векторной булевой функции введем новую одномерную булеву функцию: пусть $\theta \in V_m$, $F(x) = (f_1(x), ..., f_m(x))$ — векторная (n, m)-функция, тогда новая булева функция
$\theta \cdot F(x) := \theta_1 f_1(x) \oplus \theta_2 f_2(x) \oplus ... \oplus \theta_m f_m(x)$.

Определение 5. *Нелинейностью векторной булевой (n, m)-функции $F(x) := (f_1(x), f_2(x), ..., f_m(x))$ от n переменных называется целое число $N_F := \min\limits_{\theta\in V_m\setminus 0^m} \{N_{\theta\cdot F}\}$, где $\theta \cdot F := \bigoplus\limits_{i=1}^{m} \theta_i f_i$.*

После данных определений мы можем поставить экстремальную задачу максимизации нелинейности векторных булевых функций:

$$N_F \to \max_F, \qquad F \in V_{2^n}^m \tag{1}$$

Кроме прямого способа посчитать нелинейность одномерной булевой функции, можно воспользоваться более удобным с вычислительной точки зрения преобразованием Уолша-Адамара:

Определение 6. *Преобразованием Уолша-Адамара булевой функции f от n переменных называется целочисленная функция на V_n, определяемая следующим равенством*
$W_f(u) := \sum_{x\in V_n} (-1)^{f(x)\oplus\langle x,u\rangle}$, *где $\langle x, u\rangle = x_1u_1 \oplus ... \oplus x_nu_n$*

Любой одномерной функции от n переменных мы можем сопоставить вектор, состоящий из коэффициентов Уолша-Адамара:

$$f \to (W_f(0), W_f(1), ..., W_f(2^n - 2), W_f(2^n - 1)),$$

который будем называть ***спектром*** Уолша-Адамара функции f.

Следующая лемма связывает нелинейность одномерной булевой функции и преобразование Уолша-Адамара:

Лемма 1. *([1]) Нелинейность булевой функции можно вычислить как $N_f = \frac{1}{2}(2^n - W_{max})$, где W_{max} максимальное по модулю значение спектра Уолша-Адамара функции f.*

Определение 7. *Спектром Уолша-Адамара для (n,m)-функции F будем называть матрицу размерности $2^m - 1 \times 2^n$ вида $\{W_{\theta\cdot F}(0), W_{\theta\cdot F}(1), ..., W_{\theta\cdot F}(2^n-1)\}_{\theta\in V_m\backslash 0^m}$, где каждая строчка спектр Уолша-Адамара для одномерной функции $\theta\cdot F$, $\theta \in V_m\backslash 0^m$*

Пример 1. *Вычислим спектр и нелинейность для векторной $(2,3)$-функции F:*

	F		$\theta\cdot F$		спектр				$N_{\theta\cdot F}$
			f_1	$1,1,0,0$	*0,*	*0,*	*-4,*	*0*	0
			f_2	$1,1,1,0$	*-2,*	*-2,*	*-2,*	*2*	1
f_1	$1,1,0,0$	$\rightarrow$	$f_1\oplus f_2$	$0,0,1,0$	$\rightarrow$ *2,*	*-2,*	*2,*	*2*	1
f_2	$1,1,1,0$		f_3	$1,1,1,1$	*-4,*	*0,*	*0,*	*0*	0
f_3	$1,1,1,1$		$f_3\oplus f_1$	$0,0,1,1$	*0,*	*0,*	*4,*	*0*	0
			$f_3\oplus f_2$	$0,0,0,1$	*2,*	*2,*	*2,*	*-2*	1
			$f_3\oplus f_2\oplus f_1$	$1,1,0,1$	*-2,*	*2,*	*-2,*	*-2*	1

Минимальное значение из столбца $N_{\theta\cdot F}$ является нелинейностью функции F по определению, значит нелинейность F равна нулю.

Можно сформулировать следующую лемму, которая вытекает из определения нелинейности, определения спектра (n,m)-функции и леммы 1:

Лемма 2. *Нелинейность векторной булевой (n,m)-функции F можно вычислить как $N_F = \frac{1}{2}(2^n - W_{max})$, где $W_{max} := \max_{u\in V_n,\theta\in V_m\backslash 0^m}(|W_{\theta\cdot F}(u)|)$ — максимальный по модулю элемент спектра Уолша-Адамара векторной функции F.*

Как видно из Леммы 2, задача максимизации нелинейности векторной булевой функции (1) эквивалентна задаче минимизации максимального по модулю значения элемента спектра Уолша-Адамара:

$$\max_{u\in V_n,\theta\in V_m\backslash 0^m}(|W_{\theta\cdot F}(u)|) \rightarrow \min_F, \qquad F\in V_{2^n}^m \tag{2}$$

Теперь обратимся к понятию сбалансированности булевой функции:

Определение 8. *Булеву функцию f от n переменных называют сбалансированной, если ее вес равен $wt(f) = 2^{n-1}$. Вес функции $wt(f)$ — количество единиц в ее векторе значений.*

Хорошо известно, что сбалансированность также можно определить с помощью преобразования Уолша-Адамара:

Лемма 3. *Булева функция сбалансирована тогда и только тогда, когда $W_f(0) = 0$.*

Определим сбалансированность векторной булевой (n,m)-функции:

Определение 9. *Векторная (n,m)-функция сбалансирована, если она принимает каждое значение в V_m одинаковое 2^{n-m} количество раз.*

Однако иногда нам будет удобно пользоваться эквивалентным определением:

Утверждение 1. *([2]) Векторная (n, m)-функция $F(x) = (f_1(x), f_2(x), ..., f_m(x))$ сбалансирована тогда и только тогда, когда каждая ненулевая линейная комбинация f_i сбалансирована или, эквивалентно, если для любой $\theta \in V_m \backslash 0^m$ функция $\theta \cdot F$ сбалансирована.*

Пользуясь Леммой 3 и Утверждением 1, запишем экстремальную задачу с учетом требования сбалансированности:

$$\max_{u \in V_n, \theta \in V_m \backslash 0^m} (|W_{\theta \cdot F}(u)|) \to \min_F, \qquad F \in V_{2^n}^m \tag{3}$$

при ограничении:

$$W_{\theta \cdot F}(0) = 0, \text{ для } \forall \theta \in V_m \backslash 0^m \tag{4}$$

Таким образом, мы поставили две задачи, задачу (2) и задачу (3)–(4), которые кратко будем называть задачей без ограничений и сбалансированной задачей соответственно. В дальнейшем мы будем решать их эвристическими методами.

2 Границы максимальной нелинейности булевой (n, m)-функции

В данной главе рассмотрим леммы, теоремы и утверждения, которые помогут нам построить оценки максимальной нелинейности (n, m)-функции. В конце главы приведем таблицу ограничений на максимальную нелинейность (n, m)-функции для различных n и m.

Определение 10. *Если $\exists$ (n, m)-функция F с нелинейностью N_F и не $\exists$ функции $G : N_G > N_F$, то величину N_F называют максимальной нелинейностью $N(m, n)$.*

2.1 Ограничения сверху

Для нелинейности произвольной булевой функции f, а значит и для произвольной векторной (n, m)-функции справедлива верхняя оценка

Лемма 4. *([2])*

$$N(n, m) \leq 2^{n-1} - 2^{(n-2)/2}, n \geq 2 \tag{5}$$

Существуют классы булевых одномерных функций, которые достигают максимума нелинейности, например, бент-функции:

Определение 11. *Бент-функция — булева функция от четного числа переменных n, достигающая максимума нелинейности $2^{n-1} - 2^{(n-2)/2}$*

Понятие бент-функции для векторных (n, m)-функций обобщается следующим образом:

Определение 12. *Векторная (n, m)-бент-функция $F = (f_1(x), f_2(x), ..., f_m(x))$ называется бент-функцией, если каждая ненулевая линейная комбинация $f_1, f_2, ..., f_m$ является бент-функцией.*

Неравенство (5) обращается в равенство при следующих условиях:

Лемма 5. *([3]) Векторная (n, m)-бент-функция существует тогда и только тогда $n \geq 2m$ и n четно. Это эквивалентно тому, что $N(n, m) = 2^{n-1} - 2^{(n-2)/2}$, если $n \geq 2m$ и n четно.*

Доказано, что приведенное ниже неравенство (6) сильнее, чем неравенство (5):

Лемма 6. *(Граница Сидельникова-Шабата-Воденая [4])*

$$N(n, m) \leq 2^{n-1} - \frac{(3 \cdot 2^n - 2 - 2\frac{(2^n-1)(2^{n-1}-1)}{2^m-1})^{1/2}}{2}, m > n - 2 \quad (6)$$

Правая часть в неравенстве (6) не обязательно целочисленная. Поэтому мы можем использовать округление в меньшую сторону в качестве оценки сверху.

В [4, 5] доказывается:

Лемма 7. *([4, 5]) $N(n, m) = 2^{n-1} - 2^{(n-1)/2}$, если n нечетно и $n = m$.*

В [6] доказан следующий факт:

Утверждение 2. *([6]) Для $n = 3, 5, 7, n > m$ верно, что $N(n, m) = 2^{n-1} - 2^{(n-1)/2}$*

В дальнейшем для того, чтобы оценить максимальную нелинейность снизу и сверху, нам понадобятся линейные двоичные коды.

Определение 13. *Линейный двоичный (n, k) код — это линейное подпространство C размерности k векторного пространства V_n.*

Векторы из C называют кодовыми словами, n — длиной слова , k — размерностью пространства кодовых слов, кодовым расстоянием $D(C)$ — минимальное расстояние между кодовыми словами.

В нашей работе мы будем использовать только двоичные коды, поэтому в дальнейшем будем просто говорить о линейных кодах без уточнения, что они являются двоичными.

Линейный код с параметрами: длиной слова n, размерностью пространства кодовых слов k, кодовым расстоянием D будем стандартно обозначать как (n,k,D)-код.

Теорема 1. *(Граница линейного кода [7]) Если ни при каком $D \geq N_F$ не существует линейного $(2^n, n + 1 + m, D)$ кода, тогда не существует (n, m)-функции с нелинейностью равной N_F.*

Чтобы применить предыдущую теорему нужно знать параметры, при которых линейных код не существует:

Теорема 2. *(Граница Хемминга [8]) Для любого кода длины n, размерности k с минимальным кодовым расстоянием D выполняется: $n - k \geq log_2(\sum_{i=0}^{\lfloor (D-1)/2 \rfloor} C_n^i)$*

Из теоремы 1 и 2 получим следствие:

Следствие 1. *Если выполняется: $2^n - (n + 1 + m) < log_2(\sum_{i=0}^{\lfloor (D-1)/2 \rfloor} C_{2^n}^i)$, то не существует (n, m)-функции с нелинейностью равной D.*

Для следующий леммы нам понадобится определение самодополняющего кода:

Определение 14. *Код называется самодополняющим, если из того, что он содержит вектор v, следует, что он содержит вектор $1^n - v$.*

Например, если самодополняющий код содержит вектор $(0, 1, 1, 1)$, он должен также содержать вектор $(1, 0, 0, 0)$.

Лемма 8. *([7]) Если не существует линейного самодополняющего $(2^n, n + 1 + m, D)$ кода с параметром $D \geq N_F$, тогда не существует (n, m)-функции с нелинейностью равной N_F.*

Для следующей теоремы введем дополнительные обозначения и понятия. Количество кодовых слов в коде C с весом i обозначим как A_i, полином Кравчука определим как $P_k(i) := \sum\limits_{0 \leq j \leq k} (-1)^j C_i^j C_{l-i}^{k-j}$, где $C_n^k := \frac{n(n-1)(n-2)\cdots(n-(k-1))}{k!}$ и $n \in \mathbb{Z}, k, i \in \mathbb{N} \cup 0$— обобщенный бином Ньютона.

Теорема 3. *(Граница линейного программирования [7]) Не существует линейного самодополняющего кода с параметрами $(l = 2^n, \lceil log_2(2 + S_{opt}) \rceil, D)$, где S_{opt} – оптимальное значение функции для следующей задачи линейного программирования:*

$$2(A_D + A_{D+1} + ... + A_{l/2-1}) + A_{l/2} \to \max$$

при ограничениях

$$\sum_{D \leq i \leq l/2-1} A_i(P_k(i) + P_k(l - i) + A_{l/2} \geq -2C_l^k$$

для четных k, $2 \leq k \leq l$

$$A_i \geq 0, D \leq i \leq l/2$$

Также отметим важное, очевидное из определения нелинейности векторной функции свойство:

Замечание 1. *При $m > 1$ имеет место неравенство $N(m, n) \leq N(m - 1, n)$*

2.2 Ограничения снизу

Приведем утверждения теоремы и леммы, которые дают оценку максимальной нелинейности снизу. Паттерсон и Видеманн в [6] показали, что верно:

Утверждение 3. *([6])Для $m = 1, n \geq 15$ верно, что $N(n, m) > 2^{n-1} - 2^{(n-1)/2}$*

В [9] доказывается:

Лемма 9. *([9]) Если n четно, $1 < n < 2m$ и $m \leq n$, то $N(n, m) \geq 2^{n-1} - 2^{n/2}$*

Известно, что любую булеву функцию можно представить в виде алгебраической нормальной формы (АНФ или полином Жегалкина) — это форма представления булевой функции в виде полинома с коэффициентами вида 0 и 1, в котором в качестве произведения используется

операция конъюнкции, а в качестве сложения — сложение по модулю 2. Степенью булевой функции называют степень данного полинома.

Векторы значений множества булевых функций от n переменных степени не больше r образуют код, который называется кодом Рида-Маллера r-го порядка длины 2^n, $RM_{r,n}$. Заметим, что код $RM_{r',n}$ включает в себя код $RM_{r,n}$, если $r' > r$, или, как мы будем говорить чаще, $RM_{r',n}$ является надкодом $RM_{r,n}$.

Теорема 4. *(Граница надкода [7]) Если линейный $(2^n, K, D)$ код является надкодом для кода Рида-Маллера первого порядка $RM_{1,n}$ и $1 \leq m \leq K - n - 1$, то существует (n, m)-функция с нелинейностью $N_f \geq D$.*

Отметим, что доказательство данной теоремы конструктивное. Известно, что код Рида-Маллера $RM_{r,n}$ имеет следующие параметры длину 2^n, размерность $S = 1 + C_n^1 + ... + C_n^r$ и минимальное кодовое расстояние 2^{n-r} [8]. С учетом этих фактов и теоремы 3 получим:

Следствие 2. *Если $1 \leq m \leq C_n^1 + ... + C_n^r - n$, то существует (n, m)-функция с нелинейностью $N_f \geq 2^{n-r}$.*

Лемма 10. *([7]) Дуальный код к расширенному БЧХ коду, $exBCH^{\perp}_{2t+1,n}$ является надкодом для $RM_{1,n}$ и имеет параметры: длину 2^n, размерность $K = tn + 1$, минимальное кодовое расстояние $D \geq 2^{n-1} - (t-1)2^{n/2}$ при $t > 1$ и $2t - 1 < 2^{\lceil n/2 \rceil} + 1$*

Из теоремы 4 и леммы 9 следует:

Следствие 3. *(n, m)-функция с нелинейностью $D \geq 2^{n-1} - (t-1)2^{n/2}$ может быть построена при $1 \leq m \leq (t-1)n$, $t > 1$ и $2t - 1 < 2^{\lceil n/2 \rceil} + 1$*

2.3 Таблица ограничений на максимальную нелинейность $N(n, m)$

На основании приведенных в главе 2.1, 2.2 лемм, теорем и утверждений построим таблицу ограничений на максимальную нелинейность. Два числа в ячеке таблицы соответствуют нижнему и верхнему ограничению на максимальную нелинейность, в случае если указано одно число — нижнее и верхнее ограничение совпадают. Каждое ограничение имеет сноску на лемму, теорему, утверждение, $\alpha, \beta, ..., \mu, \nu$, с помощью которого оно было получено. Подобная таблица для $3 \leq n \leq 8, 1 \leq m \leq 8$ была построена в статье [7]. Мы достроили столбец для $m = 9$ и строчку для $n = 9$. Отметим, что при желании данную таблицу можно продолжить для больших значений n и m.

α: Лемма 4.

β: Лемма 5.

γ: Лемма 6 (Граница Сидельникова-Шабата-Воденая).

δ: Лемма 7.

ϵ: Утверждение 2.

ζ: Теорема 1 (Граница линейного кода).

η: Лемма 8, Теорема 3 (Граница линейного программирования).

Таблица 1: Максимальная нелинейность $N(n,m)$

n\m	1	2	3	4	5	6	7	8	9
3	2^{ϵ}	2^{ϵ}	$2^{\kappa,\alpha}$	$1^{\kappa,\gamma}$	-	-	-	-	-
4	6^{β}	6^{β}	4^{κ}-5^{α}	$4^{\kappa,\zeta}$	$4^{\kappa,\zeta}$	$4^{\kappa,\zeta}$	$2^{\kappa,\zeta}$	$2^{\kappa,\zeta}$	$2^{\kappa,\zeta}$
5	12^{ϵ}	12^{ϵ}	12^{ϵ}	12^{ϵ}	12^{δ}	8^{κ}-10^{ζ}	8^{κ}-10^{ζ}	8^{κ}-9^{ζ}	8^{κ}-9^{ζ}
6	28^{β}	28^{β}	28^{β}	24^{ι}-28^{α}	24^{ι}-26^{ζ}	24^{λ}-26^{γ}	24^{μ}-25^{γ}	$24^{\mu,\gamma}$	$24^{\mu,\gamma}$
7	56^{ϵ}	56^{ϵ}	56^{ϵ}	56^{ϵ}	56^{ϵ}	56^{ϵ}	56^{δ}	48^{μ}-54^{η}	48^{μ}-54^{η}
8	120^{β}	120^{β}	120^{β}	120^{β}	112^{λ}-120^{α}	112^{λ}-120^{α}	112^{λ}-120^{α}	112^{λ}-116^{γ}	96^{λ}-115^{γ}
9	$\frac{242^{\xi}-}{244^{\alpha}}$	$\frac{240^{\nu}-}{244^{\alpha}}$	$\frac{240^{\nu}-}{244^{\alpha}}$	$\frac{240^{\nu}-}{244^{\alpha}}$	$\frac{240^{\nu}-}{244^{\alpha}}$	$\frac{240^{\nu}-}{244^{\alpha}}$	$\frac{240^{\nu}-}{244^{\alpha}}$	$\frac{240^{\nu}-}{244^{\alpha}}$	240^{δ}

ι: Лемма 9.

κ: Теорема 4 (Граница надкода), Следствие 2 (Коды Риды-Маллера).

λ: Теорема 4 (Граница надкода), Лемма 10, Следствие 3 (БЧХ-коды).

μ: авторы статьи [7] используют некоторую дополнительную информацию о кодовом расстоянии БЧХ-кодов.

ν: Замечание 1.

ξ: достигнуто эвристическим алгоритмом среди некоторых специальных классов функций [10],[11].

Почти все приведенные в данной главе оценки максимальной нелинейности были запрограммированы на языке Java, Теорема 3 вычислялась на Matlab.

3 Описания используемых алгоритмов поиска

В данной главе опишем эвристические алгоритмы — алгоритм имитации отжига, алгоритм восхождения на вершину и генетический алгоритм — с помощью которых будем решать задачу без ограничений (2) и сбалансированную задачу (3)–(4). Каждый алгоритм различается в зависимости от решаемой задачи, поэтому реализован в двух вариантах, с учетом и без учета требования сбалансированности.

3.1 Алгоритм имитации отжига

Имитация отжига — это метод оптимизации, использующая упорядоченный случайный поиск на основе аналогии с процессом образования в веществе кристалической структуры с минимальной энергией при охлаждении.

Преимуществом имитации отжига является свойство избегать локальных минимумов оптимизируемой функции, и продолжить поиск глобального минимума. Это достигается за счет принятия не только изменений, уменьшающих значение целевой функции, но и некоторых изменений, увеличивающих ее значение, в зависимости от температуры T — характеристики моделируемого процесса. Чем выше температура, тем больше вероятность ухудшающего изменения [12].

Пример применения имитации отжига к данной задаче можно найти в статье [13].

Управляемые параметры: α — параметр уменьшения температуры, it — количество итераций на каждом уровне температуры, k — количество изменяемых координат на каждой итерации, b — количество запретов для поиска с запертом (опционно).

3.1.1 Алгоритм имитации отжига для задачи без ограничений

1. Инициализация

   - Определим функцию изменения температуры $T(t)$ как $T(t) = \alpha \cdot t$, где $\alpha \in (0, 1)$.
   - Зададим минимальную температуру $t_{\min}$ и начальную температуру $t_{\max}$, так чтобы переход к функции, у которой нелинейность меньше, осуществлялся в наихудшем случае (случай, который максимально увеличивает нелинейность) с вероятностями 10^{-5} и 0.95 соответственно.
   - Зададим или случайно сгенерируем (n, m)-функцию F_{glob}, также введем (n, m)-функцию F, так что $F = F_{glob}$.
   - $t = t_{\max}$.

2. Пока $t > t_{\min}$

   - Пусть it — количество итераций на каждом уровне температуры. Для каждого фиксированного уровня температуры t выполним it-раз следующие шаги:
     (a) Случайным образом изменим k координат (не исключена возможность повторного изменения уже измененной координаты) в матрице (n, m)-функции F.
     (b) Если значение нелинейности (n, m)-функции F повысилось или уменьшилось число $W_{\max}$ в матрице спектра Уолша-Адамара, оставим данное изменение в (n, m)-функции F.
     (c) В противном случае оставим изменения с вероятностью $P(\Delta_W) = e^{(-\Delta_{WH}/2-1)/t_i}$, где $\Delta_W = W'_{\max} - W_{\max}$, $W'_{\max}$ – максимальное по модулю значение Уолша-Адамара для измененной функции, а $W_{\max}$ – изначальной.
   - Если значение нелинейности (n, m)-функции F стало больше значения нелинейности (n, m)-функции F_{glob}, изменим функцию $F_{glob} = F$.
   - Понизим температуру $t = T(t)$

3.1.2 Алгоритм имитации отжига для сбалансированной задачи

Чтобы учесть требования сбалансированности, сгенерируем (n, m)-функцию F_{glob} так, чтобы она была сбалансирована. И каждый раз вместо изменения координат матрицы (n, m)-функции, будем переставлять ее столбцы. В остальном алгоритм остается таким же.

3.1.3 Модификации

Возможной модификацией алгоритма является добавление поиска с запретами. Пусть b – число запретов, тогда после изменения какой-либо координаты или столбца (при поиске сбалансированных функций) запретим изменять ее/его в течении b последующих итераций.

3.2 Алгоритм восхождения на вершину

Восхождение на вершину — это метод локальной оптимизации. Является итеративным алгоритмом, который начинает поиск со случайного решения и пытается найти в его определенной окрестности решение лучше. Как только решение лучше найдено, алгоритм переходит в него. Таким образом, процесс повторяется пока ни одного улучшения не может быть найдено.

Пример применения алгоритма восхождения на вершину для поиска булевых функций с заданными свойствами (в том числе высокой нелинейности) можно найти в статьях [14], [15].

Введем понятие k-окрестности функции:

Определение 15. *k-окрестностью (n, m)-функции F будем называть множество (n, m)-функций, матричные представления которых отличаются не более чем на k позиций от матричного представления функции F.*

Для сбалансированных функций:

Определение 16. *k-сбалансированной окрестностью (n, m)-функции F будем называть множество (n, m)-функций, которые получены из функции F перестановкой не более k столбцов ее матрицы.*

Управляемые параметры: k — параметр, определяющий окрестность функции.

3.2.1 Алгоритм восхождения на вершину для задачи без ограничений

1. Зададим или случайно сгенерируем (n, m)-функцию F.
2. Будем перебирать функции из k-окрестности (n, m)-функции F.
3. Если в процессе перебора найдем функцию, у которой нелинейность выше чем у функции F или при одинаковом значении нелинейности в матрице Уолша-Адамара меньше количество максимальных по модулю значений $W_{\max}$, примем ее за новую функцию F. Перейдем к пункту 2

Заметим, что алгоритм остановится, когда мы переберем все функции из k-окрестности и ни при одной из них не произойдет улучшение (n, m)-функции F. Алгоритм завершит свою работу, так как нелинейность функции ограничена сверху.

3.2.2 Алгоритм восхождения на вершину для сбалансированной задачи

Чтобы учесть требования сбалансированности изменим первые два пункта на следующие:

1. Зададим или случайно сгенерируем сбалансированную (n, m)-функцию F.

2. Будем перебирать функции из k-сбалансированной окрестности (n, m)-функции F.

Очевидно, что каждая функцию из k-сбалансированной окрестности сбалансированной функции сбалансирована. Поэтому на выходе мы получим сбалансированную функцию.

3.3 Генетический алгоритм

Генетический алгоритм — это метод оптимизации, который использует механизмы, аналогичные естественному отбору в природе. Он состоит из процессов отбора, скрещивания и мутации, которые обычно взаимодействуют следующим образом: случайно создаётся множество начальной популяции, к особям которого применяются операции скрещивания и мутации. В результате чего получается множество новых решений. Затем производится отбор лучших (с точки зрения целевой функции) решений в следующее поколение. Этот набор действий повторяется итеративно.

Применительно к задаче без ограничений был применен встроенный генетический алгоритм Matlab, а также запрограммированны варианты генетического алгоритма, приведенные ниже.

3.3.1 Генетический алгоритм для задачи без ограничений

Управляемые параметры: N — размер популяции, it — количество итераций, α — доля мутантов в поколении, k — количество генов (элементов матрицы), изменяющихся у мутантов, β — доля отбора с возможностью повторения отбираемых функций (опционно).

В генетическом алгоритме набор решений (в нашем случае (n, m)-функций) изменяется с помощью схемы скрещивания, мутации и отбора, чтобы улучшить характеристики элементов множества решений.

В качестве целевой функции (в генетических алгоритмах ее часто называют функций приспособленности) примем нелинейность (n, m)-функции.

1. Случайно сгенерируем N (n, m)-функций — первоначальное множество родителей.

2. Скрещивание. Обозначим матрицы родительских функций b_1, b_2, а матрицу ребенка b_c. В качестве генов рассмотрим ячейки матрицы функции. Если родительские гены совпадают, т.е. $b_1[i][j] = b_2[i][j]$, то и ген ребенка будет равен родительскому $b_c[i][j] = b_1[i][j] = b_2[i][j]$. В противном случае (когда $b_1[i][j] \neq b_2[i][j]$), значение $b_c[i][j]$ случайно выберем между 0 и 1 с равномерной вероятностью. Скрещивая все возможные пары, породим $\frac{N(N-1)}{2}$ потомков. Добавим родительские функции в множество потомков, чья мощность станет равна $\frac{N(N+1)}{2}$.

3. Мутация. Из множества потомков выберем $\alpha \cdot \frac{N(N+1)}{2}$ особей, где α - доля мутантов в поколении. У каждой выбранной функции случайно выберем и изменим k ячеек матрицы.

4. Отберем из множества потомков N (n, m)-функции с наибольшей нелинейностью (возможны повторы) и примем отобранные функции за новое множество родителей. В случае совпадения значения нелинейности в первую очередь будем отбирать те особи, у которых наименьшее количество максимальных значений Уолша-Адамара.

5. Будем выполнять действия 2-4, пока не наступит условие остановки (количество итераций it).

3.3.2 Генетический алгоритм для сбалансированной задачи

Вариант генетического алгоритма для сбалансированных функций с некоторыми модификациями был взят из статьи [16].

Управляемые параметры: N — размер популяции, it — количество итераций.

1. Генерация первоначального множества (n, m)-функций. Заведем массив, содержащий 2^{n-m} копии всех возможных 2^m векторов размерности m. Тогда, случайно переставляя элементы массива получим N случайных сбалансированных (n, m)-функций.

2. Скрещивание-мутирование.

   Обозначим матрицы родителей $b_1(x)$, $b_2(x)$, а матрицу ребенка $b_c(x)$. Тогда под $b_i(x_j)$ будем понимать столбец матрицы функции. Также количество появлений $b_i(x_j), i = 1, 2$ в $b_c(x)$ обозначим $\#(b_i(x_j))$. Опишем алгоритм скрешивания-мутирования функций $b_1(x)$, $b_2(x)$:

   Случай $b_1(x_j) = b_2(x_j)$

   - Если $\#(b_1(x_j)) < 2^{n-m}$, то $b_c(x_j) = b_1(x_j)$, увеличиваем $\#(b_1(x_j))$ на единицу.
   - в противном случае $b_c(x_j) = b_1(x_k)$ $k \neq j$, такое что $\#(b_1(x_k))$ наименьшее. Увеличим $\#(b_1(x_k))$ на единицу.

   Случай $b_1(x_j) \neq b_2(x_j)$

   - Если $\#(b_1(x_j)) = 2^{n-m}$ и $\#(b_2(x_j)) = 2^{n-m}$, то $b_c(x_j) = b_1(x_k)$ $k \neq j$, такое что $\#(b_1(x_k))$ наименьшее. Увеличим $\#(b_1(x_k))$ на единицу.
   - Если $\#(b_1(x_j)) < \#(b_2(x_j))$, то $b_c(x_j) = b_1(x_j)$ и $\#(b_1(x_j))$ увеличим на единицу.
   - Если $\#(b_1(x_j)) > \#(b_2(x_j))$, то $b_c(x_j) = b_2(x_j)$ и $\#(b_2(x_j))$ увеличим на единицу.
   - Если $\#(b_1(x_j)) = \#(b_2(x_j))$ и $\#(b_i(x_j)) < 2^{n-m}$, то $b_c(x_j)$ случайно выбираем между $b_1(x_j)$ и $(b_2(x_j))$. Увеличим соответствующий $b_i(x_j)$ на единицу. Данный алгоритм включает в себя как скрещивание, так и мутацию. Обычно $b_c(x_j) = b_1(x_j)$ или $b_2(x_j)$. Однако возникают ситуации, когда это не выполняется, т.е. $b_c(x_j)$ – мутация. Алгоритм гарантирует, что ребенок получится сбалансированным. Скрещивая все возможные пары, породим $\frac{N(N-1)}{2}$ потомков. Добавим родительские функции в множество потомков, чья мощность станет равна $\frac{N(N+1)}{2}$.

3. Отберем из множества потомков N (n, m)-функции с наибольшей нелинейностью (возможны повторы) и примем отобранные функции за новое множество родителей. В случае совпадения значения нелинейности в первую очередь будем отбирать те особи, у которых наименьшее количество максимальных значений Уолша-Адамара.

4. Будем выполнять действия 2-3, пока не наступит условие остановки (количество итераций it).

3.3.3 Модификации

1. Возможной модификацией алгоритма является процедура встряхивания (reset), которая при отсутствии улучшения нелинейности на протяжении s итераций заново генерирует $N - 2$ функций родительского пула, оставляя в нем только 2 функции с наибольшей нелинейностью.

2. Второй возможной модификацией является модификация отбора функций. Вместо отбора, который не учитывает повторения отбираемых функций, будем отбирать $\lfloor \beta \cdot N \rfloor$ $\beta \in [0, 1]$ (n, m)-функций также с возможностью повтора, а оставшиеся $\lceil (1 - \beta) \cdot N \rceil$ (n, m)-функций отберем, требуя, чтобы они были различными. Если в процессе отбора потомки закончатся, случайно сгенерируем недостающие особи. Заметим, что если параметр $\beta = 0$, то все отобранные особи будут различными, если $\beta = 1$, алгоритм отбора будет таким же, как до модификации.

3.4 Общая модификация для всех алгоритмов

Любую одномерную булеву функцию от n переменных можно представить не только в виде 0-1 вектора значений длины 2^n, но также в виде 0-1 вектора такой же длины, где каждый элемент вектора обозначает коэффициент при определенном мономе полинома Жегалкина (АНФ представление функции). Отметим, что от вектора значений булевой функции к векторному АНФ представлению можно легко перейти с помощью преобразования Мёбиуса [17].

Из этого следует, что также любую векторную булеву функцию $F = (f_1(x), ..., f_m(x))$ можно представить в виде матрицы АНФ представления, где каждая i строчка вектор АНФ преставления булевой функции f_i. Если применить к данному представлению описанные выше алгоритмы — получим новую версию используемых алгоритмов.

Данную модификацию будем кратко называть модификацией Мёбиуса.

4 Результаты работы алгоритмов

Модификация Мёбиуса для всех алгоритмов дает результат хуже, чем стандартная версия, для всех исследуемых алгоритмов. Добавление поиска с запретами в алгоритм имитации отжига и процедуры встряхивания в генетический алгоритм может как улучшить, так и ухудшить достигаемую алгоритмами нелинейность, поэтому от них также было решено отказаться.

Модификация метода отбора функций в генетическом алгоритме при параметре β — доля отбора с возможностью повторения отбираемых функций — ориентировочно лежащем в отрезке $[0.8, 0.9]$ для несбалансированного поиска и близкого к нулю для сбалансированного часто улучшает или по крайней мере не ухудшает достигаемую алгоритмом нелинейность. Поэтому данную модификацию решено было оставить.

Генетический алгоритм Matlab всегда дает заметно худший результат, по сравнению со всеми алгоритмами, приведенные в главе 3. Например, для $n = 9$, $m = 2$ алгоритм Matlab (при размере и числе популяций, равным 100) находит значение нелинейности не более 228, тогда как все исследуемые алгоритмы дают нелинейность не меньше 232 (Таблица 6). Поэтому он был исключен из рассмотрения.

В дальнейшем в данной главе проанализируем работу алгоритмов для конкретных значений n и m и сравним их по следующим параметрам:

- max_N_F — максимальной достигнутой алгоритмом нелинейностью,
- $num\_max\_N_F$ — числу функций с максимально-достигнутой нелинейностью;
- EN_F — средней нелинейности;
- $Etime$ — среднему времени работы алгоритма (в секундах). Все программы были написаны на одном языке программирования (Java) и исполнялись на одном компьютере.

Как правило размер тестовой статистики будем брать равным 100.

Отметим, что при увеличение некоторых управляемых параметров (количества итераций, размера популяции в генетическом, k-окрестности в восхождении на вершину) результат работы алгоритмов по первым трем критериям только улучшится, однако время работы алгоритмов может существенно возрасти. Поэтому главным критерием сравнения (с учетом поставленных в работе задач) будет служить максимальная достигнутая и средняя нелинейность при разумном времени работы алгоритма.

Кроме того, важно знать, насколько лучше работают алгоритмы по сравнению с поиском нужной функции с помощью случайной генерации — случайно сгенерируем функцию и примем ее за искомую векторную функцию F, только если нелинейность вновь сгенерированной функции выше, чем у F. В качестве управляемого параметра случайная генерация принимает только время своей работы, которое будет равно $Etime$.

Также добавим в рассматриваемые алгоритмы комбинации описанных в главе 3 алгоритмов:

- Имитация отжига+восхождения на вершину — к полученной с помощью имитацией отжига функции применяется алгоритм восхождения на вершину
- Генетический+восхождение на вершину — к полученной с помощью генетического алгоритма функции применяется алгоритм восхождения на вершину

4.1 Результат работы алгоритмов для задачи без ограничений

Так как одной из задач является уточнение границ максимальной нелинейности, будем стараться рассматривать такие n и m, для которых нижние и верхние границы максимальной нелинейности не совпадают. И если какой-то исследуемый алгоритм найдет булеву функцию с нелинейностью выше, чем ее нижняя граница, то мы сможем говорить, что мы улучшили границу максимальной нелинейности снизу.

Сравним работу алгоритмов для задачи максимизации нелинейности без ограничений на сбалансированность для случаев $(n = 5, m = 6)$, $(n = 4, m = 3)$, $(n = 6, m = 4)$ и $(n = 8, m = 5)$.

В Таблице 2 приведена статистика работы алгоритмов в пространстве $(5, 6)$-функций со следующими управляемыми параметрами:

- Алгоритм имитации отжига: $\alpha = 0.9$ — параметр уменьшения температуры, $it = 1000$ — количество итераций на каждом уровне температуры, $k = 1$ — количество изменяемых координат на каждой итерации.
- Алгоритм восхождения на вершину: $k = 2$ — параметр, определяющий окрестность функции.
- Генетический алгоритм: $N = 50$ — размер популяции, $it = 100$ — количество итераций, $\alpha = 0.4$ — доля мутантов в поколении, $k = 2$ — количество генов, изменяющихся у мутантов, $\beta = 0.9$ — доля отбора с возможностью повторения отбираемых функций.

Таблица 2: Результат работы алгоритмов при $n = 5, m = 6$, размер тестовой статистики 100

Алгоритмы	max_N_F	$num\_max\_N_F$	EN_F	$Etime$
Имитация отжига	9	93	8.93	8.74
Восхождение на вершину	10	1	8.98	5.18
Генетический алгоритм	10	1	8.95	21.54
Имитации отжига + восхождение на вершину	10	2	9.02	12.59
Генетический + восхождение на вершину	10	4	9.04	26.26
Поиск случайной генерацией	9	1	8.01	10

Вывод: Для $n = 5, m = 6$ максимальная нелинейность $N(5, 6) = 8 - 10$ (Таблица 1). Однако мы можем видеть из таблицы 2, что все алгоритмы в данном случае улучшили границу максимальной нелинейности снизу (нашли функцию с большей нелинейностью), и почти все улучшили так, что она стала равна границе максимальной нелинейности сверху. То есть в данной случае нам удалось уточнить границу максимальной нелинейности снизу и теперь $N(5, 6) = 10$. Кроме того, заметим, что все исследуемые алгоритмы работают лучше, чем поиск случайной генерацией.

В Таблице 3 приведена статистика работы алгоритмов в пространстве $(4, 3)$-функций со следующими управляемыми параметрами:

- Алгоритм имитации отжига: $\alpha = 0.95$ — параметр уменьшения температуры, $it = 1000$ — количество итераций на каждом уровне температуры, $k = 1$ — количество изменяемых координат на каждой итерации.

- Алгоритм восхождения на вершину: $k = 3$ — параметр, определяющий окрестность функции.

- Генетический алгоритм: $N = 100$ — размер популяции, $it = 100$ — количество итераций, $\alpha = 0.4$ — доля мутантов в поколении, $k = 2$ — количество генов, изменяющихся у мутантов, $\beta = 0.8$ — доля отбора с возможностью повторения отбираемых функций.

Таблица 3: Результат работы алгоритмов при $n = 4, m = 3$, размер тестовой статистики 100

Алгоритмы	max_N_F	$num\_max\_N_F$	EN_F	$Etime$
Имитация отжига	4	100	4	0.78
Восхождение на вершину	4	100	4	0.09
Генетический алгоритм	4	100	4	4.5
Имитации отжига + восхождение на вершину	4	100	4	0.86
Генетический + восхождение на вершину	4	100	4	4.6
Поиск случайной генерацией	4	100	4	4

Вывод: Для $n = 4, m = 3$ максимальная нелинейность $N(4, 3) = 4 - 5$ (Таблица 1). В данном случае мы не смогли улучшить границу максимальной нелинейности, и как видно из таблицы 3, любой исследуемый алгоритм работает не лучше, чем поиск случайной генерацией.

В Таблице 4 приведена статистика работы алгоритмов в пространстве $(6, 4)$-функций со следующими управляемыми параметрами:

- Алгоритм имитации отжига: $\alpha = 0.95$ — параметр уменьшения температуры, $it = 2100$ — количество итераций на каждом уровне температуры, $k = 1$ — количество изменяемых координат на каждой итерации.

- Алгоритм восхождения на вершину: $k = 2$ — параметр, определяющий окрестность функции.

- Генетический алгоритм: $N = 100$ — размер популяции, $it = 100$ — количество итераций, $\alpha = 0.4$ — доля мутантов в поколении, $k = 2$ — количество генов, изменяющихся у мутантов, $\beta = 0.8$ — доля отбора с возможностью повторения отбираемых функций.

Таблица 4: Результат работы алгоритмов при $n = 6, m = 4$, размер тестовой статистики 100

Алгоритмы	max_N_F	$num\_max\_N_F$	EN_F	$Etime$
Имитация отжига	24	70	23.7	9.73
Восхождение на вершину	24	86	23.84	6.51
Генетический алгоритм	24	85	23.85	41.8
Имитации отжига + восхождение на вершину	24	97	23.97	15.84
Генетический + восхождение на вершину	24	98	23.98	47.09
Поиск случайной генерацией	23	1	22.01	15

Вывод: Для $n = 6, m = 4$ максимальная нелинейность $N(6, 4) = 24 - 28$ (Таблица 1). В данном случае (Таблица 4) нам не удалось улучшить границы максимальной нелинейности, однако все алгоритмы достигли нижней границы и работают лучше, чем поиск случайной генерацией.

В Таблице 5 приведена статистика работы алгоритмов в пространстве $(8, 5)$-функций со следующими управляемыми параметрами:

- Алгоритм имитации отжига: $\alpha = 0.95$ — параметр уменьшения температуры, $it = 3000$ — количество итераций на каждом уровне температуры, $k = 1$ — количество изменяемых координат на каждой итерации.
- Алгоритм восхождения на вершину: $k = 2$ — параметр, определяющий окрестность функции.
- Генетический алгоритм: $N = 100$ — размер популяции, $it = 100$ — количество итераций, $\alpha = 0.4$ — доля мутантов в поколении, $k = 1$ — количество генов, изменяющихся у мутантов, $\beta = 0.9$ — доля отбора с возможностью повторения отбираемых функций.

Размер тестовой статистики был сокращен до 20 из-за продолжительного времени работы алгоритмов.

Таблица 5: Результат работы алгоритмов при $n = 8, m = 5$, размер тестовой статистики 20

Алгоритмы	max_N_F	$num\_max\_N_F$	EN_F	$Etime$
Имитация отжига	107	9	106.45	256.95
Восхождение на вершину	108	15	107.65	1615.02
Генетический алгоритм	108	1	105.5	384.61
Имитации отжига + восхождение на вершину	108	14	107.65	1485.17
Генетический + восхождение на вершину	108	15	107.7	1900.58
Поиск случайной генерацией	103	20	103	250

Вывод: Для $n = 8, m = 5$ максимальная нелинейность $N(8, 5) = 112 - 120$ (Таблица 1). В данном случае (Таблица 5) ни один из алгоритмов не достиг нижней границы максимальной нелинейности. Все алгоритмы работают лучше, чем поиск случайной генерацией. Среди чистых алгоритмов выделяется алгоритм восхождения на вершину, который при данных управляемых параметрах находит функции лучше, чем отжиг и генетический, и не хуже, чем комбинации алгоритмов. При этом, однако, его время работы одно из самых продолжительных.

В Таблице 6 приведена статистика работы алгоритмов в пространстве $(9, 2)$-функций со следующими управляемыми параметрами:

- Алгоритм имитации отжига: $\alpha = 0.95$ — параметр уменьшения температуры, $it = 3000$ — количество итераций на каждом уровне температуры, $k = 1$ — количество перестановок столбцов на каждой итерации.
- Алгоритм восхождения на вершину: $k = 2$ — параметр, определяющий окрестность функции.
- Генетический алгоритм: $N = 100$ — размер популяции, $it = 100$ — количество итераций, $\alpha = 0.4$ — доля мутантов в поколении, $k = 1$ — количество генов, изменяющихся у мутантов $\beta = 0.9$ — доля отбора с возможностью повторения отбираемых функций.

Таблица 6: Результат работы алгоритмов при $n = 9, m = 2$, размер тестовой статистики 100

Алгоритмы	max_N_F	$num\_max\_N_F$	EN_F	$Etime$
Имитация отжига	232	75	231.7	52.55
Восхождение на вершину	234	59	233.22	119.78
Генетический алгоритм	234	3	232.11	81.22
Имитации отжига + восхождение на вершину	234	77	233.73	152
Генетический + восхождение на вершину	236	1	233.62	193.27
Поиск случайной генерацией	227	4	226.04	100

Вывод: Для $n = 9, m = 2$ максимальная нелинейность $N(8, 5) = 240 - 244$ (Таблица 1). Все алгоритмы работают лучше, чем поиск случайной генерацией. Среди чистых алгоритмов положительно выделяется алгоритм восхождения на вершину. Лучшую статистику (за исключением времени работы) имеют комбинированные алгоритмы. (Таблица 6).

Также отметим, что для случая $n = 8, m = 9$ исследуемыми алгоритмами удалось получить функцию с нелинейностью 100, тогда как теоретическая максимальная нелинейность $N(8, 9) = 96 - 115$.

Почти во всех рассмотренных случаях исследуемые алгоритмы работают лучше, чем поиск случайной генерацией. Комбинация алгоритмов обладают лучшей статистикой по функциям, но имеют наибольшее время работы. Среди чистых алгоритмов лучше других ищет функции алгоритм восхождения на вершину.

4.2 Результат работы алгоритмов для сбалансированной задачи

Заметим, что для сбалансированной векторной булевой (n, m)-функции по определению должно выполняться $n \geq m$. Поэтому сравним работу алгоритмов для задачи максимизации нелинейности векторной булевой функции с ограничением на сбалансированность для следующих случаев $(n = 5, m = 5)$, $(n = 6, m = 4)$, $(n = 8, m = 5)$ и $(n = 9, m = 2)$.

Также отметим, что для сбалансированного случая не ставится задача улучшить границу максимальной нелинейности снизу или даже достичь ее, так как условие сбалансированности серьезно ограничивает поиск, и максимальная нелинейность для сбалансированных функций может быть ниже, чем максимальная нелинейность произвольной булевой функции.

В Таблице 7 приведена статистика работы алгоритмов в пространстве $(6, 4)$-функций со следующими управляемыми параметрами:

- Алгоритм имитации отжига: $\alpha = 0.95$ — параметр уменьшения температуры, $it = 2000$ — количество итераций на каждом уровне температуры, $k = 2$ — количество перестановок столбцов на каждой итерации.
- Алгоритм восхождения на вершину: $k = 2$ — параметр, определяющий окрестность функции.
- Генетический алгоритм: $N = 100$ — размер популяции, $it = 100$ — количество итераций, $\beta = 0.1$ — доля отбора с возможностью повторения отбираемых функций.

Таблица 7: Результат работы алгоритмов при $n = 6, m = 4$, сбалансированная задача, размер тестовой статистики 100

Алгоритмы	max_N_F	$num\_max\_N_F$	EN_F	$Etime$
Имитация отжига	22	100	22	17.86
Восхождение на вершину	24	27	22.54	0.24
Генетический алгоритм	22	100	22	31.9
Имитации отжига + восхождение на вершину	24	63	23.26	24.59
Генетический + восхождение на вершину	24	57	23.14	38.77
Поиск случайной генерацией	22	100	22	20

Вывод: Для $n = 6, m = 4$ максимальная нелинейность $N(6, 4) = 24 - 28$ (Таблица 1). В сбалансированном случае для $n = 6, m = 4$ имитация отжига и генетический алгоритм работают не лучше, чем поиск случайной генерацией. Остальные алгоритмы достигли нижней границы максимальной нелинейности. (Таблица 7).

В Таблице 8 приведена статистика работы алгоритмов в пространстве $(5, 5)$-функций со следующими управляемыми параметрами:

- Алгоритм имитации отжига: $\alpha = 0.95$ — параметр уменьшения температуры, $it = 1000$ — количество итераций на каждом уровне температуры, $k = 1$ — количество перестановок столбцов на каждой итерации.

- Алгоритм восхождения на вершину: $k = 2$ — параметр, определяющий окрестность функции.

- Генетический алгоритм: $N = 80$ — размер популяции, $it = 100$ — количество итераций, $\beta = 0$ — доля отбора с возможностью повторения отбираемых функций.

Таблица 8: Результат работы алгоритмов при $n = 5, m = 5$, сбалансированная задача, размер тестовой статистики 100

Алгоритмы	max_N_F	$num\_max\_N_F$	EN_F	$Etime$
Имитация отжига	10	98	9.96	9.59
Восхождение на вершину	10	85	9.7	0.07
Генетический алгоритм	10	43	8.86	20.8
Имитации отжига + восхождение на вершину	10	100	10	10.55
Генетический + восхождение на вершину	10	84	9.68	20.86
Поиск случайной генерацией	10	20	8.4	20

Вывод: Для $n = 5, m = 5$ максимальная нелинейность $N(6, 4) = 12$ (Таблица 1). Все исследуемые алгоритмы работают лучше, чем поиск случайной генерацией. Среди чистых алгоритмов при данных управляемых параметрах можно выделить алгоритм имитации отжига. (Таблица 8).

В Таблице 9 приведена статистика работы алгоритмов в пространстве $(8, 5)$-функций со следующими управляемыми параметрами:

- Алгоритм имитации отжига: $\alpha = 0.95$ — параметр уменьшения температуры, $it = 3000$ — количество итераций на каждом уровне температуры, $k = 1$ — количество перестановок столбцов на каждой итерации.

- Алгоритм восхождения на вершину: $k = 2$ — параметр, определяющий окрестность функции.

- Генетический алгоритм: $N = 100$ — размер популяции, $it = 100$ — количество итераций, $\beta = 0.1$ — доля отбора с возможностью повторения отбираемых функций.

Таблица 9: Результат работы алгоритмов при $n = 8, m = 5$, сбалансированная задача, размер тестовой статистики 20

Алгоритмы	max_N_F	$num\_max\_N_F$	EN_F	$Etime$
Имитация отжига	106	20	106	250.17
Восхождение на вершину	108	8	106.6	52.87
Генетический алгоритм	106	1	104	288.32
Имитации отжига + восхождение на вершину	108	8	106.8	302.37
Генетический + восхождение на вершину	108	7	106.7	346.96
Поиск случайной генерацией	104	6	102.6	250

Вывод: Для $n = 8, m = 5$ максимальная нелинейность $N(8, 5) = 112 - 120$ (Таблица 1). Все алгоритмы работают лучше, чем поиск случайной генерацией. Алгоритм восхождения на вершину работает лучше чем чистые алгоритмы, и не хуже, чем комбинация алгоритмов, при значительно меньшем времени работы. (Таблица 9).

В Таблице 10 приведена статистика работы алгоритмов в пространстве $(9, 2)$-функций со следующими управляемыми параметрами:

- Алгоритм имитации отжига: $\alpha = 0.95$ — параметр уменьшения температуры, $it = 3000$ — количество итераций на каждом уровне температуры, $k = 1$ — количество перестановок столбцов на каждой итерации.
- Алгоритм восхождения на вершину: $k = 2$ — параметр, определяющий окрестность функции.
- Генетический алгоритм: $N = 100$ — размер популяции, $it = 100$ — количество итераций, $\beta = 0.1$ — доля отбора с возможностью повторения отбираемых функций.

Таблица 10: Результат работы алгоритмов при $n = 9, m = 2$, сбалансированная задача, размер тестовой статистики 100

Алгоритмы	max_N_F	$num\_max\_N_F$	EN_F	$Etime$
Имитация отжига	232	74	231.48	50.42
Восхождение на вершину	236	2	232.42	39.75
Генетический алгоритм	230	2	227.51	73.51
Имитации отжига + восхождение на вершину	234	29	232.58	91.78
Генетический + восхождение на вершину	234	18	232.55	116.19
Поиск случайной генерацией	226	100	226	80

Вывод: Для $n = 9, m = 2$ максимальная нелинейность $N(8, 5) = 240 - 244$ (Таблица 1). Все алгоритмы работают лучше, чем поиск случайной генерацией. Лучшую статистику имеет алгоритм восхождения на вершину. (Таблица 10).

Все алгоритмы в рассмотренных случаях работают лучше, чем поиск случайной генерацией. Комбинации алгоритмов, как и в задаче без ограничений, обладает лучшей статистикой по функциям, но имеют наибольшее время работы. Также выделим алгоритм восхождения на вершину, который для сбалансированной задачи всегда работает в несколько раз быстрее по сравнению с другими алгоритмами и имеет статистику по функциям в трех случаях из четырех не хуже, чем комбинации алгоритмов.

Заключение

К задачам поиска векторных булевых функций с высокой нелинейностью и поиска сбалансированных векторных булевых функций с высокой нелинейностью были применены эвристические алгоритмы — алгоритм имитации отжига, алгоритм восхождения на вершину, генетический алгоритм и их комбинации. Все алгоритмы были запрограммированы на языке Java. Кроме того, к данным задачам был применен встроенный генетический алгоритм Matlab.

Ко всем алгоритмам был предложен и протестирован набор модификаций, после чего было принято решение о применении или неприменении каждой каждой модификации.

Чтобы оценить работу запрограммированных алгоритмов относительно максимально возможного значения нелинейности, были изучены теоретические границы максимальной нелинейности.

С помощью исследуемых алгоритмов для случаев $n = 5$, $m = 6$ и $n = 8$, $m = 9$ удалось уточнить теоретические границы максимальной нелинейности снизу.

По работе алгоритмов для обеих задач максимизации нелинейности векторной булевой функции, с учетом требования сбалансированности и без, была собрана статистика для различных n и m. По ней можно сделать вывод, что лучшей статистикой по найденным функциям обладают комбинированные методы поиска, комбинации генетического алгоритма с восхождением на вершину и имитации отжига с восхождением на вершину. Однако, как стоило ожидать, комбинации алгоритмов имеют наибольшее время работы.

Также выделим алгоритм восхождения на вершину, который для задачи без учета сбалансированности имеет лучшую статистику по функциям среди некомбинированных алгоритмов. Для задачи с учетом сбалансированности восхождение на вершину всегда работает в несколько раз быстрее других алгоритмов и часто ищет функции не хуже, чем комбинированные методы поиска.

Список литературы

[1] Ященко В.В. Логачев О.А., Сальников А.А. *Булевы функции в теории кодирования и криптологии.* М.:МЦНМО, 2004.

[2] Carlet C. *Boolean Methods and Models*, chapter Vectorial boolean functions for cryptography. Cambridge University Press, 2008.

[3] Nyberg K. Perfect nonlinear s-boxes. In *Advances in Cryptology—EUROCRYPT'91*, pages 378–386. Springer, 1991.

[4] Chabaud F. and Vaudenay S. Links between differential and linear cryptanalysis. In *Advances in Cryptology—EUROCRYPT'94*, pages 356–365. Springer, 1994.

[5] Nyberg K. Differentially uniform mappings for cryptography. In *Advances in cryptology—Eurocrypt'93*, pages 55–64. Springer, 1993.

[6] Patterson N. J. and Wiedemann D. H. The covering radius of the $(2^{15}, 16)$ reed-muller code is at least 16276. *Information Theory, IEEE Transactions on*, 29(3):354–356, 1983.

[7] Wadayama T., Hada T., Wakasugi K., and Kasahara M. Upper and lower bounds on maximum nonlinearity of n-input m-output boolean function. *Designs, Codes and Cryptography*, 23(1):23–34, 2001.

[8] Слоэн Н. Дж. А. Мак-Вильямс Ф. Дж. *Теория кодов, исправляющих ошибки.* М: Связь, 1979.

[9] Nyberg K. S-boxes and round functions with controllable linearity and differential uniformity. In *Fast Software Encryption*, pages 111–130. Springer, 1994.

[10] Selçuk Kavut and Melek Diker Yücel. Generalized rotation symmetric and dihedral symmetric boolean functions- 9 variable boolean functions with nonlinearity 242. In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, pages 321–329. Springer, 2007.

[11] Maitra S. Boolean functions on odd number of variables having nonlinearity greater than the bent concatenation bound. *Boolean Functions in Cryptology and Information Security (NATO ASI Zvenigorod, 2007)*, pages 173–182, 2008.

[12] Лопатин А. Метод отжига. *Стохастическая оптимизация в информатике*, 1:133–149, 2005.

[13] Clark J. A ., Jacob J. L., and Stepney S. The design of s-boxes by simulated annealing. *New Generation Computing*, 23(3):219–231, 2005.

[14] Millan W., Clark A., and Dawson E. Boolean function design using hill climbing methods. In *Information Security and Privacy*, pages 1–11. Springer, 1999.

[15] Millan W., Clark A., and Dawson E. Smart hill climbing finds better boolean functions. In *In Workshop on Selected Areas in Cryptology 1997, Workshop Record*, pages 50–63, 1997.

[16] Millan W., Burnett L, Carter G., Clark A., and Dawson E. Evolutionary heuristics for finding cryptographically strong s-boxes. In *Information and Communication Security*, pages 263–274. Springer, 1999.

[17] Агафонова И.В. Алгебраическая нормальная форма булевой функции и бинарное преобразование Мёбиуса, 2013.

[18] Millan W. How to improve the nonlinearity of bijective s-boxes. In *Information Security and Privacy, Third Australasian Conference, ACISP'98*, 1998.

[19] Токарева Н. Н. Нелинейные булевы функции: бент-функции и их обобщения. *Издательство LAP LAMBERT Academic Publishing (Saarbrucken, Germany)*, 2011.

[20] Millan W., Clark A., and Dawson E. Heuristic design of cryptographically strong balanced boolean functions. In *EUROCRYPT 98, LNCS 1403*, pages 489–499. Springer-Verlag, 1998.